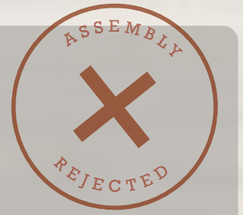
3 MIXIN

application.css

```
.btn-a {
  background: #777;
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
}
.btn-b {
  background: #ff0;
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
}
```

repeating properties in each declaration



3.1 Mixin Setup + Use

Mixins

Blocks of reusable code that take optional arguments:

_buttons.scss

```
@mixin button {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
}
```


_buttons.scss

```scss
@mixin button {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
}
.btn-a {
  @include button;
  background: #777;
}
.btn-b {
  @include button;
  background: #ff0;
}
```

application.css

```css
.btn-a {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
  background: #777;
}
.btn-b {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
  background: #ff0;
}
```

3.1 Mixin Setup + Use

Assembly Tip

Make sure the `@mixin` block comes before the `@include`, especially when importing files containing mixins.

Assembly Tip

`@include` = use a mixin

`@import` = import a file

_buttons.scss

```scss
@mixin button {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
}
.btn-a {
  @include button;
  background: #777;
}
.btn-b {
  @include button;
  background: #ff0;
}
```

application.css

```css
.btn-a {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
  background: #777;
}
.btn-b {
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
  background: #ff0;
}
```

ASSEMBLY REJECTED

repeating properties in each declaration

3.1 Mixin Setup + Use

We're Just Repeating Properties

It's more efficient to use CSS here (for now):

application.css

```
.btn-a,
.btn-b {
  background: #777;
  border: 1px solid #ccc;
  font-size: 1em;
  text-transform: uppercase;
}
.btn-b {
  background: #ff0;
}
```

If that's the case, what are mixins good for then?

application.css

```
.content {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
  border: 1px solid #ccc;
  padding: 20px;
}
```

writing three mostly-identical properties gets old

3.2 Adding Arguments

application.scss

```scss
@mixin box-sizing {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
.content {
  @include box-sizing;
  border: 1px solid #ccc;
  padding: 20px;
}
```

application.css

```css
.content {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
  border: 1px solid #ccc;
  padding: 20px;
}
```

still just copying
unchanging properties

3.2 Adding Arguments

Arguments

Values passed into a mixin, potentially altering output:

application.scss

```
@mixin box-sizing($x) {
  -webkit-box-sizing: $x;
  -moz-box-sizing: $x;
  box-sizing: $x;
}
```


application.scss

```scss
@mixin box-sizing($x) {
  -webkit-box-sizing: $x;
  -moz-box-sizing: $x;
  box-sizing: $x;
}
.content {
  @include box-sizing(border-box);
  border: 1px solid #ccc;
  padding: 20px;
}
.callout {
  @include box-sizing(content-box);
}
```

application.css

```css
.content {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
  border: 1px solid #ccc;
  padding: 20px;
}
.callout {
  -webkit-box-sizing: content-box;
  -moz-box-sizing: content-box;
  box-sizing: content-box;
}
```

3.2 Adding Arguments

Default Values

Optionally, what arguments will default to if not included:

application.scss

```
@mixin box-sizing($x: border-box) {
  -webkit-box-sizing: $x;
  -moz-box-sizing: $x;
  box-sizing: $x;
}
```


application.scss

```scss
@mixin box-sizing($x: border-box) {
  -webkit-box-sizing: $x;
  -moz-box-sizing: $x;
  box-sizing: $x;
}
.content {
  @include box-sizing;
  border: 1px solid #ccc;
  padding: 20px;
}
.callout {
  @include box-sizing(content-box);
}
```

border-box, if no argument is passed

application.css

```css
.content {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
  border: 1px solid #ccc;
  padding: 20px;
}
.callout {
  -webkit-box-sizing: content-box;
  -moz-box-sizing: content-box;
  box-sizing: content-box;
}
```

3.2 Adding Arguments

_buttons.scss

```scss
@mixin button($radius, $color) {
  border-radius: $radius;
  color: $color;
}
.btn-a {
  @include button(4px, #000);
}
```

arguments are comma-separated and passed in order

application.css

```css
.btn-a {
  border-radius: 4px;
  color: #000;
}
```

3.3 Multiple Arguments

_buttons.scss

```
@mixin button($radius, $color) {
  border-radius: $radius;
  color: $color;
}
.btn-a {
  @include button(4px);
}
```

too few arguments

application.css

```
Syntax error: Mixin button is
missing argument $color.
```

3.3 Multiple Arguments

_buttons.scss

```scss
@mixin button($radius, $color: #000) {
  border-radius: $radius;
  color: $color;
}
.btn-a {
  @include button(4px);
}
```

optional second argument

application.css

```css
.btn-a {
  border-radius: 4px;
  color: #000;
}
```

3.3 Multiple Arguments

_buttons.scss

```scss
@mixin button($color: #000, $radius) {
  border-radius: $radius;
  color: $color;
}
.btn-a {
  @include button(4px);
}
```

optionals come last

application.css

```
Syntax error: Required argument
$color must come before any
optional arguments.
```

3.3 Multiple Arguments

_buttons.scss

```
@mixin button($radius, $color: #000) {
  border-radius: $radius;
  color: $color;
}
.btn-a {
  @include button($color: #777777,
$radius: 5px);
}
```

keyword arguments allow passing in any order

application.css

```
.btn-a {
  border-radius: 5px;
  color: #777777;
}
```

3.3 Multiple Arguments

application.css

```
.btn-a {
  -webkit-transition: color 0.3s ease-in, background 0.5s ease-out;
  -moz-transition: color 0.3s ease-in, background 0.5s ease-out;
  transition: color 0.3s ease-in, background 0.5s ease-out;
}
```

commas can naturally occur in CSS values

3.4 Variable Arguments

Passing valid, comma-separated CSS as a single value:

_buttons.scss

```
@mixin transition($val) {
  -webkit-transition: $val;
  -moz-transition: $val;
  transition: $val;
}
.btn-a {
  @include transition(color 0.3s
ease-in, background 0.5s ease-out);
}
```

application.css

```
Mixin transition takes 1
argument but 2 were passed.
```


Adding `...` to an argument creates a **variable argument** (vararg):

_buttons.scss

```
@mixin transition($val...) {
  -webkit-transition: $val;
  -moz-transition: $val;
  transition: $val;
}
.btn-a {
  @include transition(color 0.3s
ease-in, background 0.5s ease-out);
}
```

application.css

```
.btn-a {
  -webkit-transition: color 0.3s
ease-in, background 0.5s ease-out;
  -moz-transition: color 0.3s
ease-in, background 0.5s ease-out;
  transition: color 0.3s ease-in,
background 0.5s ease-out;
}
```

Variable arguments in reverse:

_buttons.scss

```
@mixin button($radius, $color) {
  border-radius: $radius;
  color: $color;
}

$properties: 4px, #000;

.btn-a {
  @include button($properties...);
}
```

passes a list which is split into arguments by the mixin

application.css

```
.btn-a {
  border-radius: 4px;
  color: #000;
}
```


_buttons.scss

```scss
@mixin highlight-t($color) {
  border-top-color: $color;
}
@mixin highlight-r($color) {
  border-right-color: $color;
}
@mixin highlight-b($color) {
  border-bottom-color: $color;
}
@mixin highlight-l($color) {
  border-left-color: $color;
}
.btn-a {
  @include highlight-r(#ff0);
}
```

application.css

```css
.btn-a {
  border-right-color: #ff0;
}
```

3.5 Interpolation + Mixins

_buttons.scss

```scss
@mixin highlight($color, $side) {
  border-#{$side}-color: $color;
}
.btn-a {
  @include highlight(#ff0, right);
}
```

application.css

```css
.btn-a {
  border-right-color: #ff0;
}
```

3.5 Interpolation + Mixins


ASSEMBLING SASS
PROPERTY OF:
CODE SCHOOL
PROJECT DATE:
SEP, 2012
STAMPED INSPECTION
NO 18
CODE SCHOOL
ASSEMBLY